NAWCWD TP 8608
NAVAIR

On Improving the Convergence of the Solution of a System of Linear Equations

by

Surendra Singh
Electrical Engineering Department
University of Tulsa, OK

Klaus Halterman
and
J. Merle Elson
Research Department

AUGUST 2005

NAVAL AIR WARFARE CENTER WEAPONS DIVISION
CHINA LAKE, CA 93555-6100

Naval Air Warfare Center Weapons Division

FOREWORD

This report details the implementation of applying the stabilized bi-conjugant gradient algorithm to solve a volume integral equation. The integral equation is a solution to Maxwell's equations and involves scattering of a plane wave from a metallic nano-cyclinder. The solution algorithm is outlined, a numerical example is given, and a FORTRAN code listing is provided. This work was done during a 2-month period at the Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, California, during June and July 2005. The Office of Naval Research – American Society of Electrical Engineers (ONR-ASEE) Fellowship Program funded this effort.

This report was reviewed for technical accuracy by Pamela L. Overfelt of the Physics and Computational Sciences Branch, Research Department, NAWCWD, Code 498100D.

Approved by R. A. NISSAN, *Head* *Research Department* 3 August 2005	Under authority of W. M. SKINNER RDML, U.S. Navy *Commander*

Released for publication by
K. HIGGINS
Director for Research and Engineering

NAWCWD Technical Publication 8608

Published by . . . . Technical Information Division
Collation . . . . .Cover, 10 leaves
First printing . . . . 20 copies

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, D.C. 20503.

1 AGENCY USE ONLY *(Leave Blank)*	2 REPORT DATE August 2005	3. REPORT TYPE AND DATES COVERED Research, June-July 2005

4. TITLE AND SUBTITLE On Improving the Convergence of the Solution of a System of Linear Equations (U)	5. FUNDING NUMBERS
6. AUTHOR(S) Surendra Singh (U. Tulsa, OK), Klaus Halterman, and J. Merle Elson	
7 PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Air Warfare Center Weapons Division China Lake, CA 93555-6100	8. PERFORMING ORGANIZATION REPORT NUMBER NAWCWD TP 8608
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 800 N Quincy Arlington, VA 22201	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

(U) The computation of electromagnetic scattering from a nanowire requires the solution of a system of linear equations of the form, $Ax = b$, where the dimension of the matrix A increases with the number of unknowns. Previously, we had implemented a bi-conjugate gradient algorithm to iteratively solve this system of equations. However, this method converges very slowly when the convergence criterion is made stringent. An improved version of the algorithm, namely stabilized bi-conjugate gradient algorithm, is implemented to overcome this drawback. The new version provides very good convergence for convergence factors up to 10^{-4} but the convergence is still slow for convergence factor in the 10^{-5} - 10^{-8} range. In order to accelerate the convergence of the solution, we make use of the Levin transform to accelerate the convergence of the solution. This transform is tested in the numerical solution of three different types of integral equations using the method of moments.

14 SUBJECT TERMS Iterative Methods, Electromagnetic Scattering, Stablized Bi-Conjugate Gradient Algorithm	15 NUMBER OF PAGES 18
	16 PRICE CODE

17 SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF ABSTRACT	19. SECURITY CLASSIFICATION OF THIS PAGE	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

CONTENTS

INTRODUCTION

The work described in this report was performed in collaboration with Professor Surendra Singh, who came to the Naval Air Warfare Center Weapons Division (NAWCWD), China Lake, California, during June and July 2005. Professor Singh, a faculty member in the Electrical Engineering Department, University of Tulsa, Tulsa, Oklahoma, was visiting as an Office of Naval Research - American Society of Electrical Engineers (ONR-ASEE) Summer Faculty Fellow. He worked in the Optics, RF, and Material Physics Section of the Physics and Computations Sciences Branch. This work involves implementing the stabilized version of the bi-conjugate gradient algorithm in order to iteratively solve a linear system of equations resulting from integral equations arising in electromagnetic scattering problems. An implementation of a convergence acceleration transform to speed up the convergence of an iterative scheme is included here as the appendix to the report.

The solution of a system of linear equations can be obtained from direct methods, such as matrix inversion, and indirect methods that include a variety of iterative schemes. The conjugate gradient method is one such iterative method that provides an additional advantage: the coefficient matrix need not be stored in its entirety in memory. This provides a much needed flexibility in very fine discretizations of the geometry under investigation without taxing the memory requirements. We implement the stabilized version of the conjugate gradient method and provide some convergence studies. Noted that the stabilized version converges very rapidly to achieve a specific precision, and then tends to oscillate if higher precision is needed. To overcome this situation, we implement a transform to accelerate the convergence of the iterative scheme. The results of applying this transform to three different types of integral equations are provided.

BI-CONJUGATE GRADIENT STABILIZED METHOD (bcg-stab)

The bcg-stab method is an iterative technique to solve a system of linear equations of the form, $Ax = b$. The matrix, A, is a known matrix of order $N\text{x}N$, where N is the number of unknowns, b is the known forcing function of length N, and x is the solution vector (length N). The method can be implemented without storing the matrix A. The algorithm can be coded such that only a row or a column of the matrix is needed at a time. This can be very helpful in cases where storing the entire matrix would pose a significant storage problem. The drawback of not storing the matrix is that it gets computed twice in each iteration of the method. So we gain advantage in storage but we pay a price in increased computation time.

Here is the pseudo-code for the bcg-stab method (Reference 1):

```
Initialization: alpha=1; rerr=100; conv=$10^{-4}$; $r_0 = b$; $v_0 = p_0 = 0$; $\rho_0 = \alpha_0 = \omega_0 = 0$;
        nitm=nunkn;
        val1=dot(b,b)
        for i = 1: nitm

if (rerr > conv)
      $\rho_i = dot(b, r_{i-1})$
      $\beta = (\rho_i / \rho_{i-1})(\alpha_{i-1} / \omega_{i-1})$
      $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$
      $v_i = Ap_i$ Note: This operation can be split up so that we use one row at a time.
      $\alpha_i = \rho_i / dot(b, v_i)$
      $s = r_{i-1} - \alpha v_i$
      $t = As$ Note: This operation can be split up so that we use one row at a time.
      $\omega_i = dot(t,s) / dot(t,t)$
      $x_i = x_{i-1} + \alpha_i p_i + \omega_i s$
      $r_i = s - \omega_i t$
      $rerr = abs(dot(r_i, r_i) / val1)$
   end
end
```

SUBROUTINE bcgstab(b,nunkns,nitm,conv,x,nit,rerr)

Input Variables

b (complex vector): Right-hand side or excitation vector of length *nunkns*.
nunkns (integer): Number of unknowns (*N*).
nitm (integer): Maximum number of iterations for bcg-stab (typically set equal to *nunkns*).
conv (real): Convergence factor to stop computation for bcg-stab (typically 10^{-4} to 10^{-8}).

Output Variables

*x (*complex): Solution vector of length *nunkns*.

nit (integer): Number of iterations taken by bcg-stab method to converge.
rerr (real): Residual error. The bcg-stab method will stop when the residual error (*rerr*) becomes less than the convergence factor (*conv*).

Lippman-Schwinger Integral Equation

We implement the bcg-stab method in the numerical solution of the Lippman-Schwinger integral equation (Reference 2):

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^{inc}(\mathbf{r}) + k_0^2 \int_S \mathbf{G}^B(\mathbf{r},\mathbf{r}') \cdot \Delta\varepsilon(\mathbf{r}')\,\mathbf{E}(\mathbf{r}')\,d\mathbf{r}' \tag{1}$$

In Equation 1, the symbols have their usual meaning: $\mathbf{r} = (x, y)$, $\mathbf{E}$ is the total electric field, $\mathbf{E}^{inc}$ is the incident electric field, $\mathbf{G}^B$ is the two-dimensional (2-D) free-space Green tensor and its explicit form is given in Reference 2, $k_0 = 2\pi/\lambda$, and $\Delta\varepsilon(\mathbf{r}) = \varepsilon - 1$, when $\mathbf{r}$ is entirely within the nanowire with permittivity ε, and 0 otherwise. The nanowire is assumed to be infinite in the z direction and all spatial variations occur in the $x-y$ plane. The integration is carried out over the cross section of the nanowire that is embedded in vacuum. We solve this equation numerically for a nanowire 30 nm in size (side dimension) illuminated by an incident field of wavelength, λ = 400 nm, with the electric field vector perpendicular to the axis of the nanowire. In this example we solved for zero absorption, that is, the imaginary part of the permittivity is zero.

Table 1 convergence comparison of bcg and the bcg-stab for two values of the convergence factor. **nrx** is a geometry discretization parameter such that the number of unknowns is **N** = 2***nrx*****nrx.** # is the number of iterations taken by the method to converge.

Table 1. Convergence Comparison, **nvx** and **N**.

Parameter	No. of unknowns	**conv** $= 10^{-4}$		**conv** $= 10^{-5}$	
nrx	**N**	**bcg no.**	**bcg-stab no.**	**bcg no**	**bcg-stab no.**
20	800	90	57	127	174
30	1800	320	37	424	571
40	3200	589	19	777	688

Figures 1 and 2 show the convergence behavior of the bcg-stab method as a function of the iterations for conv = 10^{-4} and conv = 10^{-5}, respectively. We see that the method converges extremely rapidly for conv = 10^{-4} in comparison to bcg, which takes 589 iterations to converge, as indicated in Table 1. The numerical results shown in Table 1 were obtained on a Pentium IV 32-bit PC. The performance and precision of the bcg-stab method may improve if the code is run on a 64-bit machine.

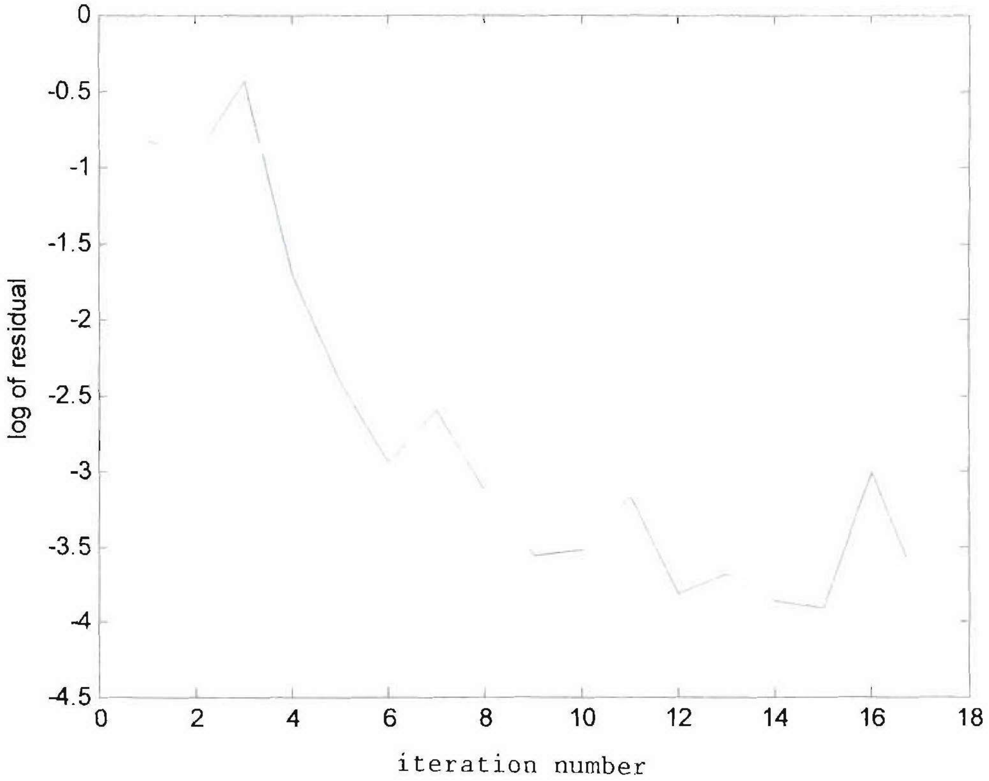

FIGURE 1. Convergence of bcg-stab Method Showing Log of Residual Versus Iterations for conv = 10^{-4} and nrx = 40.

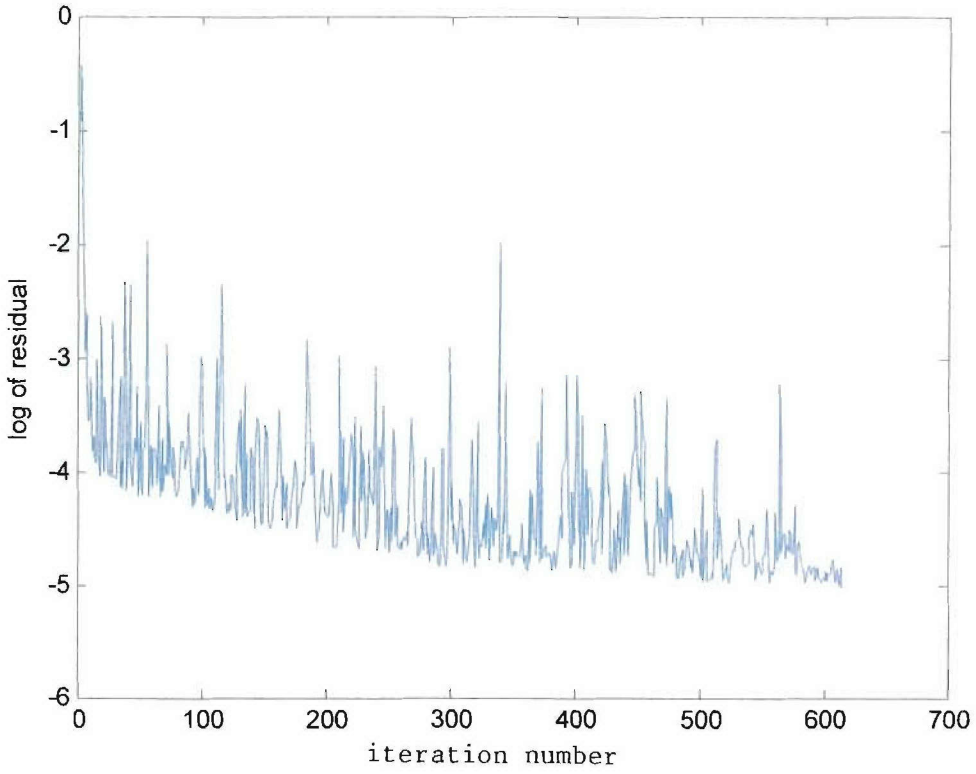

FIGURE 2. Convergence of bcg-stab Method Showing Log of Residual Versus Iterations for conv = 10^{-5} and nrx = 40.

When the imaginary part of permittivity is non-zero, the convergence of the algorithms is quite rapid. Table 2 is convergence comparison bcg and bcg (bcg_stab) as a function of wavelength, λ, and convergence factor (conv). The results are obtained from FORTRAN code: eigen2. The silver nanowire is solid and the the imaginary part of epsilon (ei) $\neq 0$, nrx = 40, (and N = 3200). The convergence of bcg_stab as a function of the iterations is shown in Figures 3 and 4.

Table1. Convergence Compairson, λ and conv.

	λ = 300 nm Number of iterations		λ = 400 nm Number of iterations		λ = 500 nm Number of iterations	
conv	bcg	bcg_stab	bcg	bcg_stab	bcg	bcg_stab
10^{-6}	8	5	66	41	152	83
10^{-8}	11	7	101	75	249	205

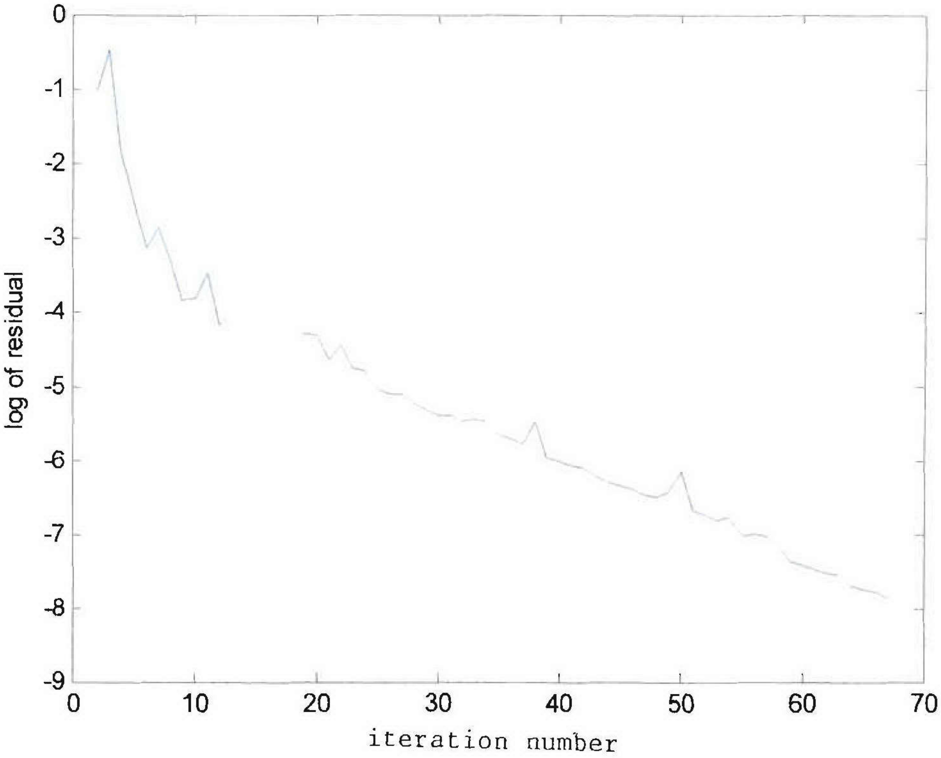

FIGURE 3. Convergence of bcg_stab Method for nrx = 40 (Number of Unknowns = 3200), λ = 400 nm, Convergence Factor = 10^{-8} (Result Obtained From MATLAB Version of the Code).

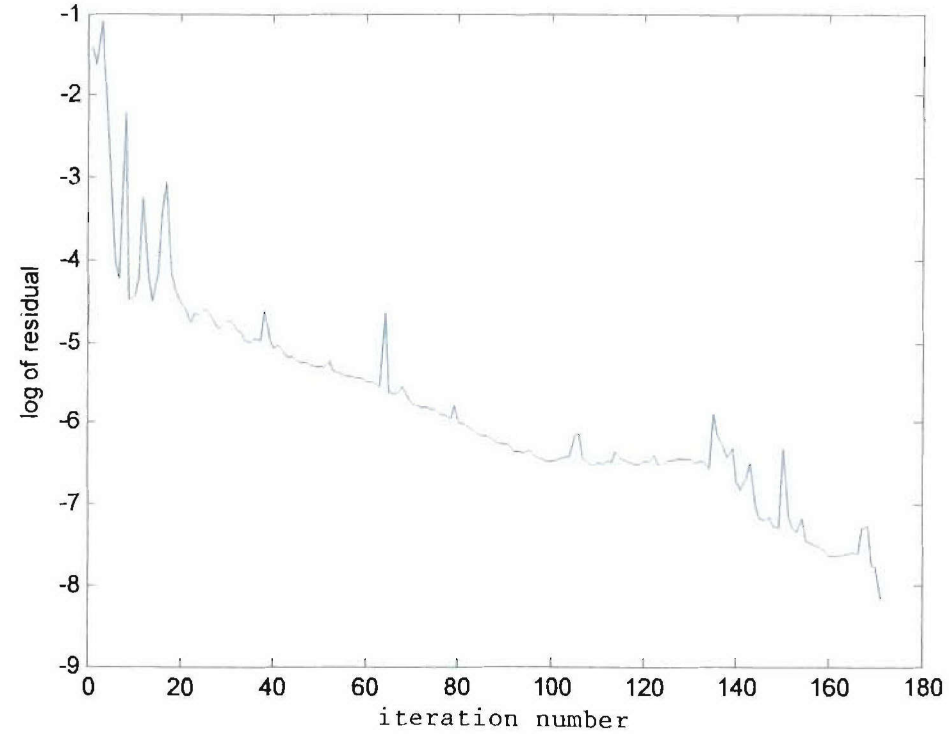

FIGURE 4. Convergence of bcg_stab Method of nrx= 40 (Number of Unknowns = 3200), λ = 500 nm, Convergence Factor (conv) = 10^{-8} (Result Obtained From MATLAB Version of bcg_stab).

LEVIN TRANSFORM

As we have seen in the results shown in Figure 2, the bcg-stab method oscillates for a long time before converging. Our goal is to apply a transform such that the sequence of vectors given by the iterative scheme (bcg-stab) will converge faster to the solution, x, of the system $Ax = b$. There are a number of convergence acceleration methods, including Levin's transform, Wynn's epsilon algorithm, Chebyschev-Toeplitz algorithm, and Brezinski's θ algorithm. Here we focus on the Levin transform that provides significant enhancement in convergence of a vector sequence. To begin with, we start with the transform as applied to a scalar sequence and then extend it to be applicable to a vector sequence. Let S_n be the partial sum of n terms of a series such that $S_n \to S$ as $n \to \infty$, where S is the sum of the series. If the partial sums are coming from an infinite series that converges extremely slowly, then the scalar sequence S_0, S_1, S_2, ..., will take a long time to converge. This is when we employ techniques to enhance the rate of convergence to arrive at S with only a small value of n. To this end, the k^{th} order Levin transform, $t_k(S_n)$, which provides an estimate of the sum of the series, S, may be computed as (Reference 3):

$$t_k^{(n)} = \frac{\sum_{i=0}^{k}(-1)^i \binom{k}{i} \left(\frac{n+i}{n+k}\right)^{(k-1)} \left(\frac{S_{n+i}}{S_{n+i+1} - S_{n+i}}\right)}{\sum_{i=0}^{k}(-1)^i \binom{k}{i} \left(\frac{n+i}{n+k}\right)^{(k-1)} \left(\frac{1}{S_{n+i+1} - S_{n+i}}\right)}, \quad k = 0,1,2, \ldots \qquad (2)$$

A significant advantage of the Levin transform is that the higher order iterates are computed from the partial sums, S_0, S_1, S_2, ..., rather than the lower iterates as in the case of Wynn's algorithm, thereby providing some immunity from the accumulation of round-off errors.

Consider the solution of the following:

$$Ax = b \qquad (3)$$

where A is a known matrix (moment) of order $N \times N$; $A = [a_{ij}]$, and x is the unknown column vector of order $N \times 1$; $x = [x_1\ x_2, \ldots x_N]^T$, and b is the known forcing function column vector of order $N \times 1$; $b = [b_1\ b_2 \ldots b_N]^T$. With A and b known, we calculate the initial estimate of x, designated as $\tilde{x}$ by the use of Gauss-Seidel relaxation, thus arriving at vectors $\{S_0\}$, $\{S_1\}$, $\{S_2\}$, and so on. Each of these vectors is of order $N \times 1$. Once we have three initial vectors, we can apply the Levin transform using Equation 1 to obtain an estimate of the solution vector, $\tilde{x}$. A numerical example of this process to a scalar sequence is given in Reference 4. The application of Equation 1 to a vector sequence requires a bit of bookkeeping, as the scalar Levin transform, $t_k^{(n)}$, now becomes a vector, $\{t_k^{(n)}\}$, of length N. As with any iterative scheme, the computation process should stop when a convergence criterion is met. We employ the following measure:

$$\frac{\|(A\tilde{x} - b)\|_2}{\|b\|_2} \le C_f \qquad (4)$$

The computations are stopped when the above criterion is met for a pre-defined convergence factor, C_f, and we have the estimate of the solution, $\tilde{x}$. We should point out that the initial vectors can be generated from a variety of schemes. It may be possible to use the iterations from a bcg algorithm and then subsequently use the Levin transform or any other vector acceleration technique to enhance the convergence. One significant advantage of using the bcg estimates is that the moment matrix, A, does not need to be stored entirely in memory, as the algorithm can be implemented in such a way that only a row or column at a time is needed in the computations. This alleviates, to a great extent, any storage problems routinely encountered when solving a very large system of equations.

NUMERICAL EXAMPLES

For our first example, we consider the solution of the integral equation for the charge distribution, $q(y)$, on a conducting strip of width = $2w$ with excitation, $V(y)$:

$$-\frac{1}{2\pi\varepsilon}\int_{-w}^{w} q(y')\ln|y-y'|\,dy' = V(y), \quad y \in (-w, w) \tag{5}$$

For illustration purposes, we take the permittivity of the medium, $\varepsilon = 1$ F/m. The integral equation is cast into a system of linear equations of the form given in Equation 2 by using method of moments (MOM) with pulse basis functions and point matching. Figure 5 shows the charge distribution for **N** = 150, and a forcing function, $V(y) = y^2$. Note that the Levin transform result matches very well with direct matrix inversion.

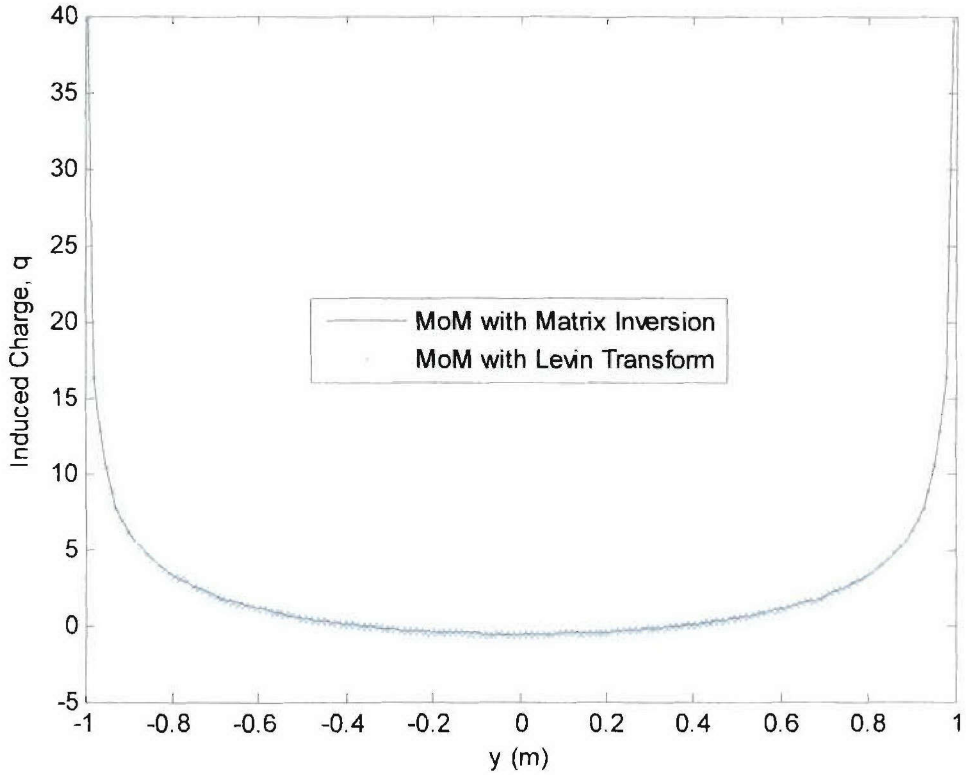

FIGURE 5. Charge Distribution on a Conducting Strip of Half-Width $w = 1$ m, Forcing Function $V(y) = y^2$, Convergence Factor $C_f = 10^{-3}$, Number of Unknowns N = 150.

The second example to illustrate the Levin transform involves a thin wire antenna. The current distribution on a thin wire can be computed by the numerical solution of Hallen's equation or Pocklington's equation. Pocklington's integral equation for the current distribution, $I(y)$, on a thin conducting wire of radius $= a$, length $= 2h$, oriented along the y - axis is given by:

$$j\frac{\eta}{4\pi k_0}\left(\frac{d^2}{dy^2} + k_0{}^2\right)\int_{-h}^{h} I(y')K(y-y')\,dy' = E_y^i(y), \quad \mathrm{y} \in (-h,h) \tag{6}$$

where

$$K(y-y')=\frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{e^{-jk_0R}}{R}d\phi' \tag{7}$$

Here the distance between the observation and source point, R, can be written as:

$$R=[(y-y')^2+4a^2\sin^2\frac{\phi'}{2}]^{1/2}. \tag{8}$$

The incident electric field is given as $E_y^i(y)=V\delta(y-y_g)$ for the antenna case, where V is the feed voltage and y_g is the feed point. In Equation 6, η is the intrinsic impedance and k_0 is the wave number of the medium. Once again, with the application of the method of moments, with suitable basis and testing functions, we can cast Equation 6 in the form of a system of linear equations: $Ax=b$. The current distribution on a center-fed, half-wavelength $(h=\lambda/4)$ thin wire antenna with $V=1$ volt, $y_g=0$m, $a=0.007022\lambda$, $C_f=10^{-3}$, and $\lambda=1$ m is shown in Figure 6. A detailed account of these examples can be found in Reference 4.

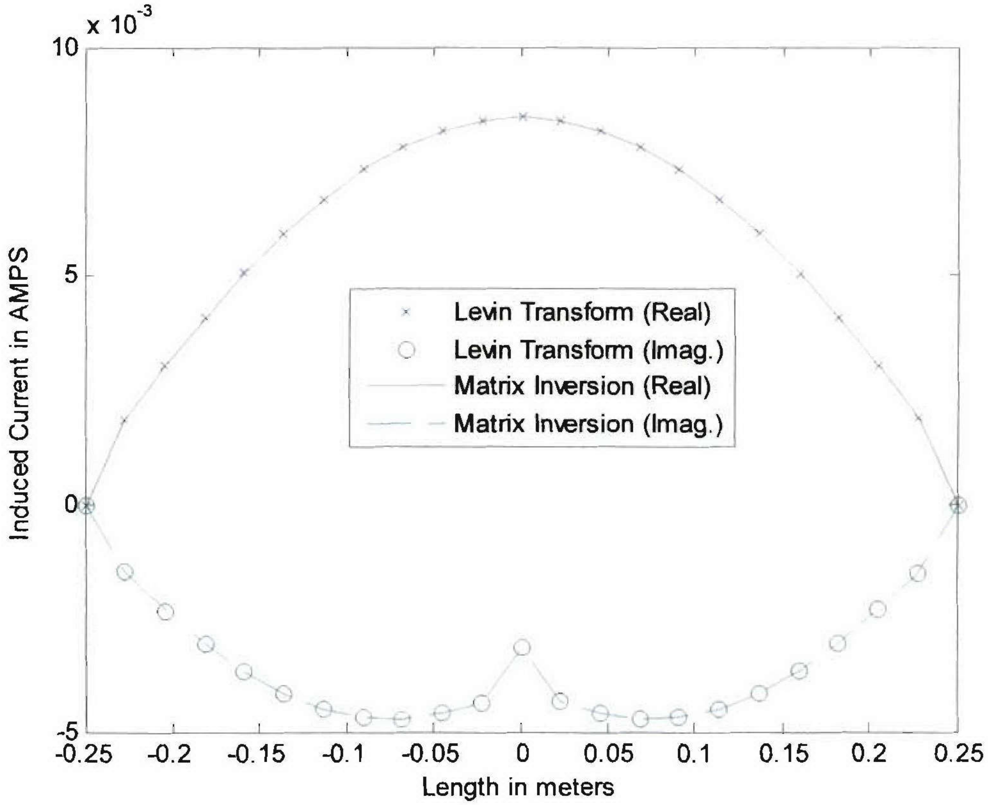

FIGURE 6. Current Distribution on a Center-Fed, Half-Wavelength, Thin Wire Antenna of Radius = 0.007022 λ, Voltage $V = 1$ Volt, Number of Unknowns $N = 21$, and Convergence Factor $C_f = 10^{-3}$.

REFERENCES

1. H. A. Van Der Vorst. "Bi-CGSTAB: A Fast and Smoothly Convergent Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comput.*, Vol. 13, No. 2 (March 1992), pp. 631-44.

2. K. Halterman, J. M. Elson, and S. Singh. "Plasmonic Resonances and Electromagnetic Forces Between Coupled Silver Nanowires," *Phys. Rev. B*, Vol. 72 (2005).

3. D. Levin. "Non-linear Transformations of Divergent and Slowly Convergent Sequences," *Math Comput.*, Vol. 16 (1973), pp. 301-22.

4. S. Singh, K. Halterman, and J. M. Elson. "An Iterative Method for the Numerical Solution of Integral Equations Using the Method of Moments," submitted to *Microwave and Opt. Tech. Lett.* (2005).

Appendix

FORTRAN CODE FOR BI-CONJUGATE GRADIENT STABILIZED METHOD

FORTRAN listing of SUBROUTINE bcg-stab:

```
subroutine bcg_stab(b,nunkns,nitm,conv,ci,nit,rerr)

 implicit none
 complex, intent(in), dimension(1:nunkns) :: b
 complex, intent(out), dimension(1:nunkns) :: ci
 integer, intent(in) :: nunkns,nitm
 integer, intent(out) :: nit
 real, intent(in) :: conv
 real, intent(out) :: rerr
 complex,dimension(1:nunkns) :: p,r,v,s,t
 complex :: beta,alpha,val1,rho,rhoi,omega
 integer ::i
 complex :: CDOTU
ci=0;p=0;v=0;alpha=1;omega=1;rho=1;r=b;s=0;t=0
rerr=100
  val1=dot_product(b,b)
 do nit=1,nitm
    if (rerr>conv) then
    rhoi=dot_product(b,r)

    beta=(rhoi/rho)*(alpha/omega)

    p=r+beta*(p-omega*v)

    do i=1,nunkns
       v(i)=CDOTU(nunkns,arow(i),1,p,1)
   enddo

    alpha=rhoi/dot_product(b,v)
   s=r-alpha*v

   do i=1,nunkns
```

```
      t(i)=CDOTU(nunkns,arow(i),1,s,1)
  enddo
  omega=dot_product(t,s)/dot_product(t,t)
  ci=ci+alpha*p+omega*s
  r=s-omega*t
  rho=rhoi
  rerr=abs(dot_product(r,r)/val1)
  ! print *,'iteration#=',nit,'residual=',rerr
  else
 exit
 end if
 end do

end subroutine bcg_stab
```